

**Amendments to the Specification:**

Please replace paragraph [0058] with the following replacement paragraph [0058]:

**[0058]** For the in-process provider of the present invention to fully enables the ADO.net programming model in the database, certain functionality must be supported by the API. Therefore, the in-process API is fully ~~symmetry~~ symmetrical with the full API as implemented by an out-of-process provider (SqlClient). Thus, for several embodiments of the present invention, one or more of the following features native to the out-of-process provider (SqlClient) are enabled:

- **MARS:** The in-process provider would support more than one pending executing command per connection. This entails the support to have multiple active stacks within a single connection. Conceptually this would enable a tree of stacks in the server within a single client side request. For this to be enabled, the top level server side frame would be assumed to have the “default” execution context, which would be cloned for each starting sub-request. Upon completion of the sub-requests, execution environment would be copied back to the “default” context, exposing semantics consistent to those exposed with top level client-side MARS. Also similar to how it is done for client-side MARS, the in-process provider’s multiple stacks will share the transaction context with other substacks.
- **Autonomous Transactions:** The API exposes the standalone concept of a transaction that can be freely associated to one or more requests that are to be executed. Based on the infrastructure provided by the unified transaction framework, the in-process provider would expose multiple top level transactions that can be associated with multiple commands at a given time.

- **Cancel / Attention:** The API exposes the ability to cancel an executing request which maps to the ability to unwind one of the possible substacks and return to the next higher CLR frame.
- **Debugging:** Hooks and debugger stops are included with the in-process provider to notify an attached debugger of the transition between TSQL and CLR frames. In this way, the in-process provider supports mixed debugging that allows end users to seamlessly step between managed code and TSQL.